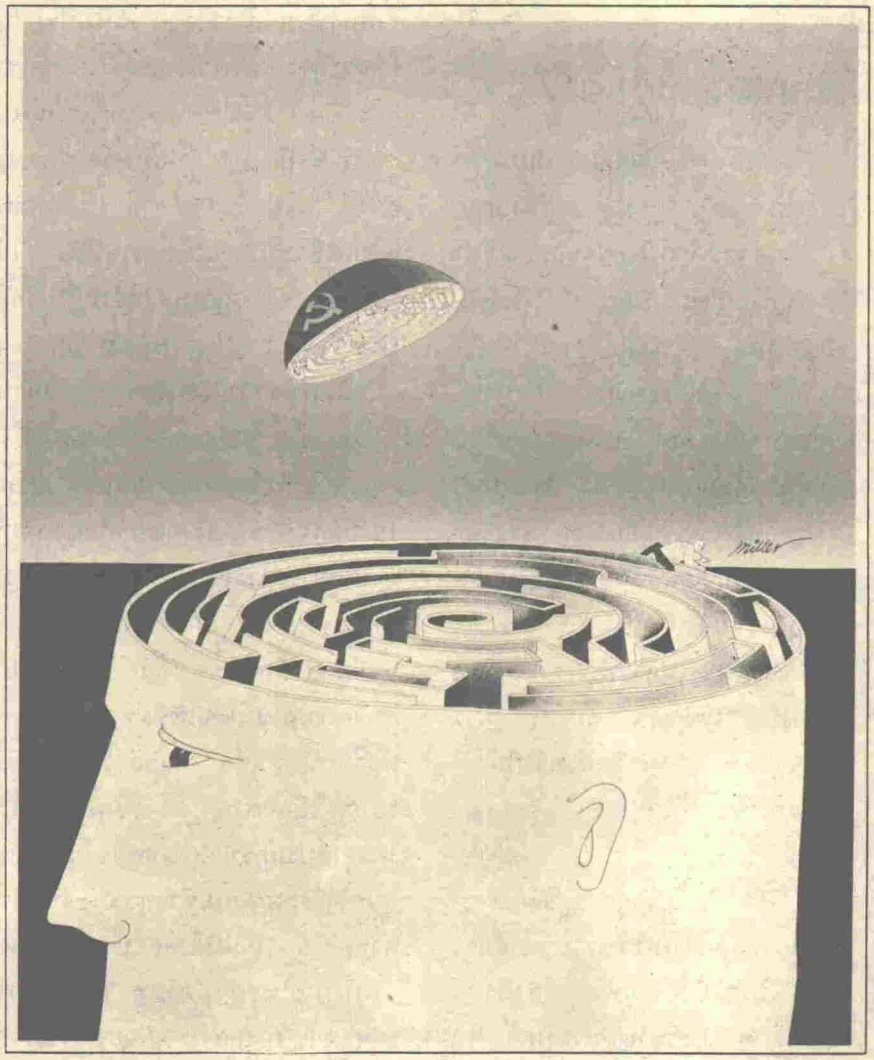
เรื่องน่าอ่าน
สุรวุฒิ สุชินโรจน์

สถาปัตยกรรม ระบบ ผู้เชี่ยวชาญ



ไขข้อข้องใจเกี่ยวกับ โครงสร้างสถาปัตย์ ในระบบผู้เชี่ยวชาญ

สถาปัตยกรรมของโปรแกรมคือโครงสร้างและหน้าที่การทำงานของส่วนต่าง ๆ ของโปรแกรม ซึ่งโดยปกติโปรแกรมชนิดที่เป็นระบบการทำงานเฉพาะด้าน ตัวอย่างเช่น โปรแกรมระบบผู้เชี่ยวชาญ (expert system) หรือโปรแกรมระบบการจัดการฐานข้อมูล (DBMS) เหล่านี้มักมีโครงสร้างที่แบ่งออกได้เป็นสองส่วนคือ ส่วนที่เป็นแกนทำหน้าที่หลักของระบบส่วนหนึ่ง และส่วนที่เป็นระบบช่วยเหลือเพื่อให้การใช้และการดูแลมีความสะดวกอีกส่วนหนึ่ง ส่วนแกนจะเป็นหัวใจของสถาปัตยกรรมของโปรแกรมแต่ละชนิด และในที่นี้จะอธิบายถึงความหมาย ที่มา และแบบการสร้าง "ส่วนแกน" ของโปรแกรมระบบผู้เชี่ยวชาญ

ประโยชน์และความสำคัญ

โปรแกรมระบบผู้เชี่ยวชาญเป็นโปรแกรมประเภทหนึ่งทางปัญญาประดิษฐ์ที่แสดงให้เห็นว่าสามารถช่วยงานประเภทที่ต้องอาศัยสติปัญญาของคนได้จริง ยกตัวอย่างเช่น โปรแกรม Prospector สำหรับช่วยระบุแหล่งแร่ที่สำคัญมูลค่ากว่า 100 ล้านเหรียญสหรัฐฯ ในปี 1980 หรือโปรแกรมมัยซิน (Mycin) นับเป็นปู่ของโปรแกรมระบบผู้เชี่ยวชาญในลักษณะที่ปรึกษา สร้างขึ้นเมื่อปี 1974 เพื่อให้คำปรึกษาข้อวินิจฉัยเกี่ยวกับชนิดของเชื้อและยาที่ใช้รักษาผู้ป่วยที่มีอาการอักเสบเนื่องจากแบคทีเรียในเลือด อาการอักเสบของเยื่อหุ้มสมองและไขสันหลัง เป็นผลงานที่รู้จักกันดี ทำให้เกิดการศึกษาสาขาคอมพิวเตอร์ทางการแพทย์ตั้งแต่นั้นมา ในปัจจุบันมีการใช้บริการโปรแกรมระบบผู้เชี่ยวชาญทางเน็ตเวอร์ก เช่น โปรแกรมแม็คซีม่า (Macsyma) โปรแกรมหาคำตอบในรูป closed form ของอัลจีบรา (เช่น หาคำตอบของสมการดิฟเฟอเรนเชียล) หรือโปรแกรมเจเนซิส (Genesis) ช่วยวิเคราะห์ดีเอ็นเอ นักวิทยาศาสตร์จำนวนหลายร้อยคนนิยมใช้บริการของทั้งสองโปรแกรมเป็นประจำ

ผลการทดสอบบางระบบได้แสดงให้เห็นความสามารถเฉพาะตัวของโปรแกรมประเภทระบบผู้เชี่ยวชาญว่า สามารถทำงานได้ดีกว่ามนุษย์ในบางเรื่อง เช่น การออกใบสั่งยาของโปรแกรมมัยซินสามารถครอบคลุมโรคได้ดีกว่าแพทย์ที่เข้าร่วมทดสอบตามคะแนนเฉลี่ย เพราะคอมพิวเตอร์มีหน่วยความจำที่สามารถบันทึกข้อมูลความรู้ได้มากกว่ามนุษย์ จึงมีโอกาสได้ใช้ความรู้และข้อมูลจำนวนมหาศาลในการคิดแก้ปัญหา ซึ่งข้อมูลเหล่านี้เป็นข้อมูลผู้ดูแลระบบและวิเคราะห์ปัญหา ได้ปรับปรุงให้ทันสมัยอยู่เสมอ

ในช่วงแรกราวปี 1975 การสร้างโปรแกรมระบบผู้เชี่ยวชาญเป็นงานใหญ่เกิดขึ้นจากฝีมือของทีมงานผู้เชี่ยวชาญทางคอมพิวเตอร์เท่านั้น ซึ่งจะใช้เวลาสร้างนานหลายปี ปัจจุบันผลงานการศึกษาวิจัยโปรแกรมระบบผู้เชี่ยวชาญเริ่มเป็นที่รู้จักกันอย่างกว้างขวาง จนกลายเป็นความรู้มาตรฐานอย่างหนึ่งทางคอมพิวเตอร์ โปรแกรมช่วยงานหลายชนิดได้รับการพัฒนาสำหรับใช้ในการสร้างระบบผู้เชี่ยวชาญ แม้แต่โปรแกรมบนไมโครคอมพิวเตอร์ ตัวอย่างเช่น โปรแกรมเขียนระบบผู้เชี่ยวชาญหรือ expert-system shell เช่น OPS5 หรือ M.1 เป็นต้น ค่าใช้จ่ายและเวลาในการสร้างโปรแกรมระบบผู้เชี่ยวชาญขนาดเล็กจะใกล้เคียงกับการเขียนโปรแกรมไมโครคอมพิวเตอร์ธรรมดาทั่วไป ดังนั้นเราจึงควรให้ความสนใจที่จะนำเทคโนโลยีทางปัญญาประดิษฐ์มาใช้ประโยชน์กับงานบางอย่าง โดยเฉพาะในพื้นที่ที่ขาดแคลนผู้เชี่ยวชาญด้านต่าง ๆ เช่น งานแพทย์พื้นบ้านในชนบท ย่อมดีกว่าปล่อยให้ชาวบ้านหาซื้อยากินเองตามมีตามเกิด หรือปรึกษาผู้ที่มิได้มีความรู้จริง งานที่ปรึกษากฎหมายหรือระเบียบวิธีการติดต่องานราชการ งานประมวลความรู้เฉพาะด้าน เช่น งานประมวลความรู้ในการจำแนกพืชสมุนไพร เพื่อใช้ในหน่วยงานที่ต้องการผู้เชี่ยวชาญในด้านนี้ กระทั่งงานภายในสำนักงานทั้งของรัฐและเอกชนที่ผู้ปฏิบัติงานต้องอาศัยคำตอบจาก "ผู้รู้"

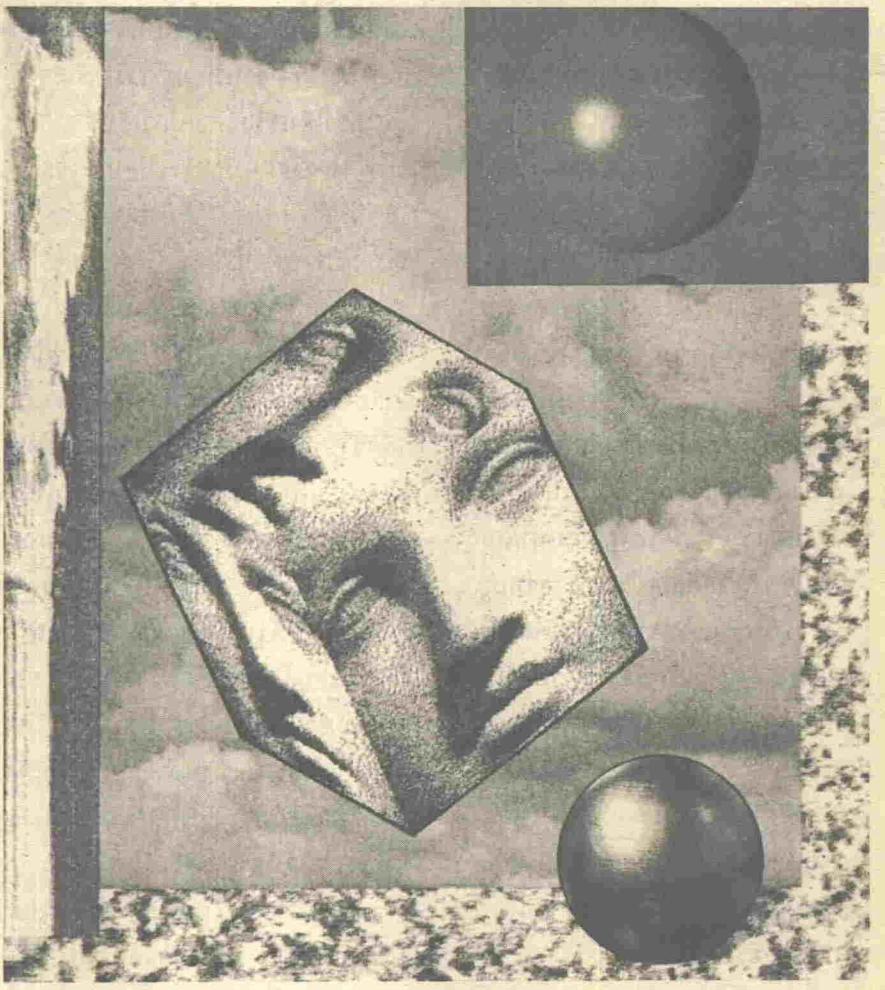
ความหมายเบื้องต้น

ระบบผู้เชี่ยวชาญเป็นหนึ่งในบรรดาโปรแกรมที่ต้องประมวลความรู้ทั้งหลาย ซึ่งเรียกตามสาขาปัญญาประดิษฐ์ว่า "ระบบที่ใช้ความรู้เป็นฐาน" (knowledge-based systems) หรือหลักในการทำงาน จึงแตกต่างจากโปรแกรมทั่วไปที่ใช้ข้อมูลเป็นหลัก บางคนอาจเรียกโปรแกรมระบบผู้เชี่ยวชาญว่า rule-based system หรือ consultation program เพื่อเน้นลักษณะพิเศษของระบบ เช่น ถ้าจะเน้นว่าเป็นโปรแกรมที่ใช้กฎ (rule) เป็นกลไกสำคัญในการทำงานก็เรียกว่า rule-based system ถ้าเน้นหน้าที่การใช้งาน เช่น โปรแกรมสำหรับให้คำปรึกษาจะเรียกว่า consultation program เป็นต้น

โดยทั่วไป โปรแกรมระบบผู้เชี่ยวชาญคือโปรแกรมที่ใช้ความรู้และการอนุมานในการแก้ปัญหา ต้องอาศัยสติปัญญาความชำนาญของมนุษย์ ในสมัยแรกความหมายกว้าง ๆ นี้ได้รวมเอาโปรแกรมที่มีสถาปัตยกรรมแตกต่างกันหลากหลาย เช่น โปรแกรมรวมวิธีการทางคณิตศาสตร์จำนวนมากและใช้เทคนิคการค้นหา (search) ทางปัญญาประดิษฐ์เป็นสำคัญ ตัวอย่างเช่น โปรแกรมแม็คซีม่านับเป็นระบบผู้เชี่ยวชาญระบบหนึ่งด้วย เนื่องจากเป็นโปรแกรมที่แก้ปัญหาเฉพาะด้านโดยผู้เชี่ยวชาญในสาขานั้น ขณะเดียวกันโปรแกรมที่ใช้วิธี plan-generate-test เช่น เดนดรอล (Dendral) เป็นโปรแกรมหาโครงสร้างของสารโดยอาศัยข้อมูลจาก mass spectrography จัดเป็นระบบผู้เชี่ยวชาญเช่นกัน ดังนั้นในทางปฏิบัติจึงไม่มีสถาปัตยกรรมใดสถาปัตยกรรมหนึ่งที่เป็นระบบผู้เชี่ยวชาญ แต่ในปัจจุบันนิยมใช้สถาปัตยกรรมแบบหนึ่งที่เรียกว่า rule-based architecture เนื่องจากเป็นแบบที่เข้าใจง่าย มีลักษณะการแสดงความรู้ในโปรแกรมที่สามารถเพิ่มเติมหรือเปลี่ยนแปลงแก้ไขภายหลัง ใช้วิธีควบคุมการทำงานโดยสร้างเป็นเชลล์ (คือเฉพาะส่วนโครงของโปรแกรม) เพื่อเป็นเครื่องมือในการสร้างโปรแกรมระบบผู้เชี่ยวชาญเฉพาะเรื่องต่อไปได้ ความสามารถในการแยกส่วนกลไกของโปรแกรมออกมาเป็นเชลล์คือสาเหตุที่ทำให้มีการสร้างโปรแกรมระบบผู้เชี่ยวชาญอื่น ๆ ตามมาเป็นจำนวนมาก ตั้งแต่ขนาดเล็กที่ประกอบด้วยความรู้ไม่กี่สิบข้อ จนถึงขนาดใหญ่ที่มีความรู้จำนวนนับร้อยนับพันข้อ ตัวอย่างสำคัญของระบบผู้เชี่ยวชาญที่สร้างขึ้นจากเชลล์ได้แก่ โปรแกรม XCON ใช้ในการจัดอุปกรณ์ระบบคอมพิวเตอร์ของบริษัทดิจิตอล อีควิปเมนต์ คอร์ปอเรชั่น เพื่อจำหน่ายเป็นชุด ๆ ตามความต้องการของลูกค้า โปรแกรม XCON สร้างจากเชลล์ชื่อ OPS5 มีความรู้ 750 ข้อ หรือโปรแกรม Drilling Advisor ของบริษัท Teknowledge Inc. ช่วยแก้ปัญหาบางอย่างในการขุดเจาะน้ำมัน สร้างจากเชลล์ชื่อ KS 300 ประกอบด้วยความรู้ 250 ข้อ ดังนั้นเมื่อเอ่ยถึงโปรแกรมระบบผู้เชี่ยวชาญจึงมักหมายถึงสถาปัตยกรรมแบบ rule-based

นอกจากนี้ความหมายของระบบ

ผู้เชี่ยวชาญในสมัยแรกยังถือความหมายของคำว่า "ผู้เชี่ยวชาญ" เป็นเรื่องสำคัญยิ่งกว่าสิ่งอื่นใด การสร้างโปรแกรมจะอาศัยการสัมภาษณ์หรือขอความรู้จากผู้เชี่ยวชาญจริง ๆ (Rich) ปัจจุบันเราพบว่าโปรแกรมที่ทำหน้าที่ให้คำปรึกษานั้นมีประโยชน์อยู่ในตัว จึงสร้างขึ้นเป็นโปรแกรม "ที่ปรึกษา" อย่างง่าย ๆ เช่น โปรแกรม TaxCut ของบริษัท Meca Software Inc. เพื่อคิดภาษีเงินได้สำหรับบุคคลทั่วไป แต่ใช้วิธีเขียนง่ายกว่าโปรแกรมที่จะกล่าวถึงในบทความนี้มาก เพราะส่วนใหญ่จะใช้วิธีที่เรียกว่า "ตารางการตัดสินใจ" (Decision Table)



ย้อนไปยังจุดกำเนิด

วิธีการใช้ความรู้และการอนุมานในโปรแกรมระบบผู้เชี่ยวชาญแบบ rule--based มีต้นกำเนิดมาจากโมเดลการใช้ปัญญาของมนุษย์ ซึ่งเป็นรูปโครงที่ช่วยแสดงให้เห็นกระบวนการที่มนุษย์ใช้ความรู้ในการคิดแก้ปัญหา ดังนั้นผู้เขียนจะอธิบายโมเดลนี้ก่อนที่จะเข้าสู่เรื่องโครงสร้างของโปรแกรมระบบผู้เชี่ยวชาญที่ถอดแบบมาจากโมเดลดังกล่าว

โมเดลการใช้ปัญญา

ความรู้ที่เรียกว่าเป็นความเชี่ยวชาญอาจแบ่งอย่างกว้าง ๆ ได้เป็นสองประเภท ประเภทแรกคือความรู้ที่ได้จากการศึกษาอย่างเป็นทางการ เช่น การเรียน การอ่าน การฝึกจากสถาบัน ประเภทที่สองคือความรู้ที่เกิดจากประสบการณ์ เป็นหลักเกณฑ์ในใจ ซึ่งอาจไม่เป็นจริงทุกครั้งหรือไม่สามารถอธิบายด้วยเหตุผลได้ทั้งหมด เรียกว่าหลักเกณฑ์กึ่งเหตุผล (heuristic) การใช้ความรู้ทั้งสองประเภทประกอบกันนี่เองที่ทำให้ผู้เชี่ยวชาญมีวิธีแก้ปัญหาที่แตกต่างจากคนธรรมดา และจะถูกเก็บไว้เป็นความทรงจำอยู่ในสมองของผู้เชี่ยวชาญ ทฤษฎีที่ให้คำอธิบายเกี่ยวกับการเก็บความรู้ในสมองของคนเราซึ่งนิยมนำมาใช้เป็นแบบอย่างในโปรแกรมทางปัญญาประดิษฐ์คือทฤษฎีที่เรียกกันว่า "ทฤษฎีความจำสัมพันธ์" (Associative Memory) หลายคนเชื่อว่ามนุษย์มีความรู้ที่เก็บไว้ในความทรงจำหลายรูปแบบ ความรู้เหล่านี้มีทั้งที่เป็นขั้นตอนการทำงาน (procedure) ข้อเท็จจริง หลักเกณฑ์ ข้อสรุปความเข้าใจในเรื่องใดเรื่องหนึ่ง เหตุและผล ฯลฯ จัดเก็บไว้ในรูปที่มีความสัมพันธ์เชื่อมโยงกันในบางลักษณะ ทำให้สามารถเรียกออกมาใช้ได้โดยการนึก ความรู้ที่อยู่ใกล้กันมากจะนึกได้เร็วกว่าความรู้ที่แยกห่างออกไป

เวลาที่เราคิด ความรู้จะถูกดึงมาใช้เป็นกลุ่ม ๆ แต่ละกลุ่มจะเป็นความรู้ที่โยงถึงกันอย่างใกล้ชิด เช่น เมื่อเรานึกถึง "โรงเรียนมัธยมที่เราเคยเรียน" เราก็จะนึกถึงครูอาจารย์เก่า ๆ สถานที่ภายในโรงเรียน และประสบการณ์เด่นที่เคยฝังใจ หากเรานึกถึง "ขนมครก" เราอาจนึกถึงกลุ่มใหม่ของความรู้ ไม่ว่าจะเป็นภาพเตาขนมครก แม่ค้าขนมครกที่เรารู้จัก เป็นต้น ซึ่งความรู้ทั้งสองกลุ่มนี้เห็นได้ชัดว่ามีความห่างไกลกัน และเกี่ยวพันกันน้อยกว่าความรู้ในกลุ่มเดียวกัน แม้ว่าจะมีส่วนที่เชื่อมโยงกันได้ทางใดทางหนึ่ง ความห่างกันของความรู้แต่ละอย่างนั้นอาจเรียงลำดับได้

แต่ยังขาดข้อสรุปในปัจจุบันว่า ความรู้ของคนเราจัดเป็นลำดับความสัมพันธ์ชิดใกล้กันได้อย่างไร

ขณะที่เราคิด การคิดจะเกิดขึ้นบริเวณส่วนหนึ่งของความจำ โดยนำความรู้ที่เกี่ยวข้องมาใช้ ประมาณกันว่าราว 6-7 กลุ่ม (Harmon) ของความรู้ส่วนใหญ่จะอยู่ตรงบริเวณความจำที่ไม่ได้ใช้ในการคิดช่วงนั้น มนุษย์จะนำความรู้ประกอบกับหลักฐานที่เกี่ยวข้องมาสร้างเป็นเหตุเป็นผล ประเมินน้ำหนักของความเชื่อ เพื่อลงความเห็นและตัดสินใจ กระบวนการในการโยงหลักฐานความรู้และการสรุปความเห็นนี้คือการ "อนุมาน" นั่นเอง การอนุมานจะเป็นลำดับขั้นโยงกันไปในลักษณะที่ต่อเนื่องเหมือนลูกโซ่ของเหตุและผลจนกระทั่งถึงเป้าหมายคือข้อสรุปสุดท้ายซึ่งแทนความเห็นของเราเกี่ยวกับเรื่องใดเรื่องหนึ่ง

การอนุมานของผู้เชี่ยวชาญเป็นเรื่องที่ค่อนข้างสลับซับซ้อน เนื่องจากในสถานการณ์ของปัญหาที่เกิดขึ้นจริง ผู้เชี่ยวชาญต้องคิดแก้ไขปัญหาภายในเวลาอันจำกัด เช่น การวินิจฉัยโรคแพทย์ไม่สามารถรอหรือหาหลักฐานข้อมูลทุกอย่างให้พร้อมก่อนได้ ดังนั้นแพทย์จึงต้องวินิจฉัยโรคทั้ง ๆ ที่ข้อมูลไม่ครบ หลักฐานบางอย่างไม่สามารถเชื่อได้ว่าเป็นจริงร้อยเปอร์เซ็นต์

ในการแก้ปัญหา คนเราจะใช้ "ความรู้" และ "วิธีการใช้ปัญญาในการแก้ปัญหา" เป็นหลัก ได้แก่ การตั้งสมมติฐาน การทดสอบสมมติฐาน เช่น การรวบรวมหลักฐาน การประเมินความเป็นไปได้ของข้อสรุปโดยอาศัยหลักฐานที่มีอยู่ และการกำจัดทางเลือกของสมมติฐานที่อ่อนเพื่อทำงานกับสมมติฐานที่เชื่อใจได้มากกว่า รวมถึงการตัดสินใจเลือกข้อสรุปที่เป็นคำตอบของปัญหา ดังนั้น "ความเชี่ยวชาญ" จึงประกอบด้วยสองสิ่งคือ "ความรู้" กับ "ความสามารถในการใช้ความรู้" นั้นเพื่อการแก้ปัญหา ทั้งสองอย่างนี้รวมกันเป็นกระบวนการใช้เหตุผลของคนเราโดยใช้กลไกหลักคือการอนุมาน

กระบวนการใช้ปัญญาของมนุษย์เป็นสิ่งที่เกิดในสมองของคนเราภายใต้ระบบการคิด ซึ่งสามารถแสดงเป็นรูปโครงสร้างส่วนประกอบสำคัญได้ดังรูปที่ 1

โครงสร้างของโปรแกรม

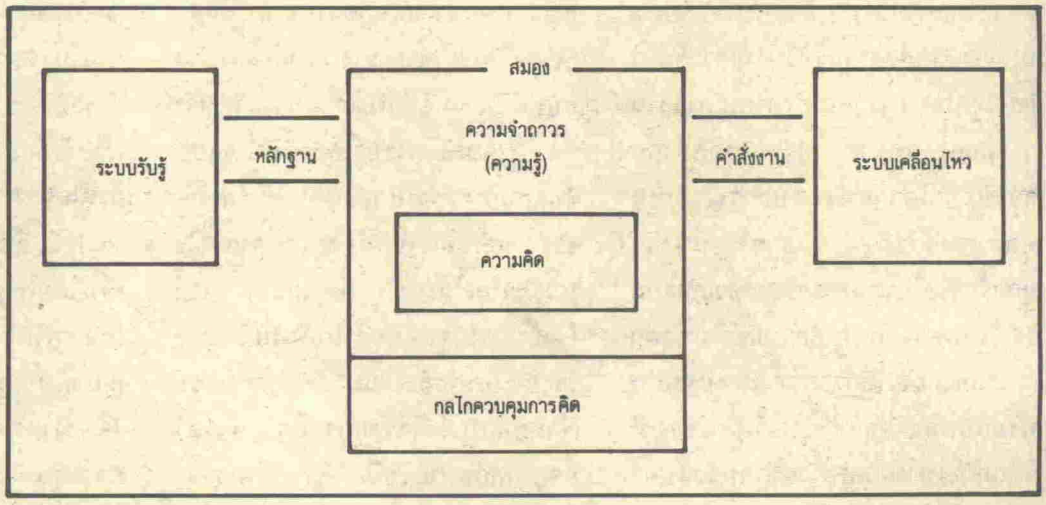
จากรูปที่ 1 โครงสร้างของโปรแกรมระบบผู้เชี่ยวชาญโดยทั่วไปประกอบด้วยส่วนสำคัญที่สุดของระบบ 3 ส่วนคือ

1. ส่วนฐานความรู้ (knowledge base) ซึ่งตรงกับส่วนความจำถาวรในสมอง ส่วนนี้จะเป็นส่วนที่เก็บความรู้ที่เป็นเหตุผลกฎเกณฑ์ที่ใช้ในการอนุมาน

2. ส่วนความจำชั่วคราว (working memory) ตรงกับส่วนของความจำที่ใช้ในการคิด ส่วนนี้จะเป็นส่วนที่ทำงานระหว่างการคิด เปรียบเสมือนสมองที่บันทึกผลการสรุปข้อเท็จจริง ตลอดจนเก็บความรู้ประเภทข้อมูลหลักฐานต่าง ๆ ในการคิด

3. ส่วนควบคุมการอนุมาน (inference engine) หรือกลไกควบคุมการคิดนั่นเอง

ในโปรแกรมคอมพิวเตอร์ทั้งสามส่วนนี้จะรวมกันเป็นแกนของระบบเรียกว่า performance system ซึ่งเราเรียกให้แตกต่างจากส่วนประกอบอื่นของระบบที่เป็นเรื่องของ human engineering นิยมเรียกกันว่า support systems ได้แก่ ส่วนเชื่อมโยงกับผู้ใช้ (user interface) ส่วนที่ช่วยในการแก้ไขปรับปรุงระบบ เช่น การเติมกฎเกณฑ์ใหม่ ๆ ให้กับฐานความรู้ (knowledge acquisition subsystem) และส่วนที่ใช้



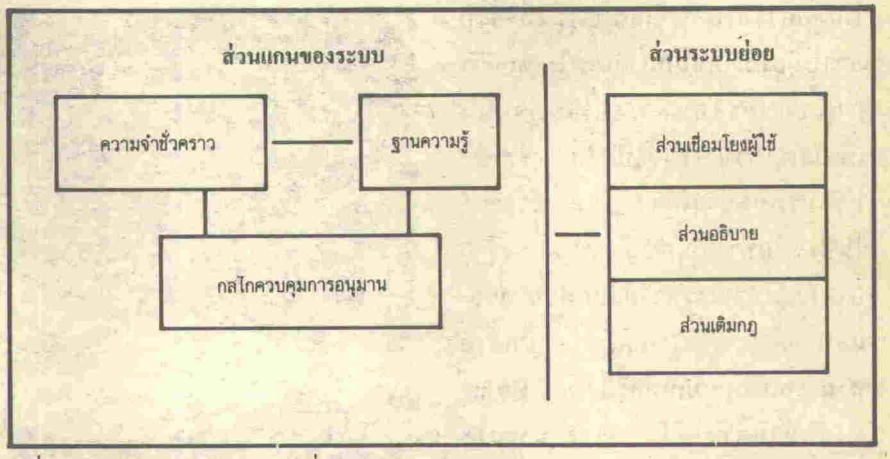
รูปที่ 1 โมเดลระบบการคิดของคน จากซ้ายไปขวาคือการนำสิ่งที่เรารับรู้มาคิดด้วยเหตุผล โดยอาศัยความรู้ภายใต้การควบคุมของกลไกบางอย่างทำให้เราคิดเป็นลำดับ ผลของการคิดคือข้อสรุปและเป็นความเข้าใจซึ่งเราใช้ในการปฏิบัติงานต่าง ๆ

ในการอธิบาย (explanation subsystem) โครงสร้างของโปรแกรมระบบผู้เชี่ยวชาญสามารถเขียนเป็นไดอะแกรมได้ดังรูปที่ 2

กลไกการทำงาน

จากโครงสร้างของระบบและแนวความคิดที่กล่าวมาจะเห็นได้ว่า โปรแกรมระบบผู้เชี่ยวชาญมีพื้นฐานแนวทางความคิดไม่เหมือนกับโปรแกรมปัญญาประดิษฐ์อื่น ๆ เพราะเป็นโปรแกรมที่เน้นการใช้ "ความรู้" ของคนเป็นตัวหลักในการแก้ปัญหาแทนที่จะเน้นความสำคัญที่เทคนิคทางปัญญาประดิษฐ์หรือการแก้ปัญหาโดยใช้อัลกอริทึม กล่าวอีกนัยหนึ่งคือ ตัวโปรแกรมของระบบผู้เชี่ยวชาญคือชุดของความรู้ที่สรุปไว้เป็นข้อ ๆ ไม่ใช่คำสั่งคอมพิวเตอร์ที่เรียงต่อ ๆ กันเป็นโปรแกรมอย่างที่เราเคยรู้จัก (ถึงแม้ในทางปฏิบัติเรายังต้องเขียนความรู้ที่ว่านี้ในรูปแบบที่บังคับไว้ จนคล้ายกับเป็นคำสั่งคอมพิวเตอร์เช่นกัน) ชุดของความรู้จะรันอยู่บนกลไก "การใช้ปัญญาในการแก้ปัญหา" อีกทีหนึ่ง ประสิทธิภาพของโปรแกรมระบบผู้เชี่ยวชาญจะดีหรือไม่ดีส่วนหนึ่งย่อมขึ้นอยู่กับ "ความรู้" ในโปรแกรม ถ้าโปรแกรม "มีความรู้ดี" ผลการทำงานก็ดี โปรแกรมมีความรู้น้อยผลการทำงานอาจเป็นที่สงสัยได้ โปรแกรมระบบผู้เชี่ยวชาญจึงมีลักษณะของการเรียนรู้ โดยผู้ใช้เป็นผู้ดูแลเพิ่มความรู้ให้กับโปรแกรมเสมอ ความรู้เก่า ๆ บางอย่างจำเป็นต้องแก้ไข ลบทิ้ง เปลี่ยนแปลงบ้าง ดังนั้นโปรแกรม "ผู้เชี่ยวชาญ" จัดว่าไม่ใช่โปรแกรมคอมพิวเตอร์ธรรมดา

กลไกการทำงานภายในของโปรแกรมระบบผู้เชี่ยวชาญจะแตกต่างจากการทำงานของโปรแกรมคอมพิวเตอร์ทั่วไป เนื่องจากกลไกหลักของการทำงานใช้วิธีการใหม่คือ กลไกของกระบวนการ



รูปที่ 2 โครงสร้างของโปรแกรมระบบผู้เชี่ยวชาญ

อนุมาน จะไม่ปฏิบัติตามคำสั่งเป็นลำดับเหมือนโปรแกรมคอมพิวเตอร์อื่น ๆ ความรู้ข้อใดข้อหนึ่งที่เหมาะสมอาจถูกนำมาใช้ในการคิดได้ทุกขณะ และต้องทำงานกับฐานข้อมูลที่เป็นความรู้ซึ่งเปลี่ยนแปลงได้ตลอดเวลาตามสถานะของการคิดภายในโปรแกรมในขณะหนึ่ง ๆ

วิธีการทำงานของโปรแกรมระบบผู้เชี่ยวชาญเป็นการคำนวณ (computation) อย่างหนึ่งเรียกว่า data driven แต่วิธีการทำงานของโปรแกรมทั่วไปเป็นการคำนวณแบบ instruction driven เรื่องนี้ผู้อ่านจะเข้าใจได้ดียิ่งขึ้นเมื่อได้เห็นผังกลไกการทำงานของระบบ

ในแง่ทฤษฎีการคำนวณ ความแตกต่างของกลไกการคำนวณในโปรแกรมคอมพิวเตอร์ทั่วไปกับกลไกการคำนวณของโปรแกรมระบบผู้เชี่ยวชาญจะมีลักษณะเดียวกับความแตกต่างระหว่างภาษาแบบ procedural เช่น ภาษาซี ภาษาปาสคาล ภาษาโคบอล เมื่อเปรียบเทียบกับภาษา declarative เช่น ภาษา LISP ภาษา Prolog ภาษา ML หมายถึงความแตกต่างโดยพื้นฐานของสถาปัตยกรรมคอมพิวเตอร์ที่เป็นแบบฟอน นอยแมน (Von Neumann) กับแบบที่ตรงกันข้าม โปรแกรมระบบผู้เชี่ยวชาญหรือโปรแกรมที่เขียนด้วยภาษา declarative มีพื้นฐานกลไกการคำนวณแบบไม่ใช่ฟอน นอยแมน โปรแกรมประเภทนี้ต้องรันบนเครื่องกล (machine) ของตัวเอง เช่น โปรแกรมภาษาโปรล็อก ซึ่งเป็นภาษาแบบ declarative จะรันบนเครื่องกลโปรล็อก (prolog machine) ส่วนโปรแกรมระบบผู้เชี่ยวชาญต้องรันบนเครื่องกลที่เรียกว่าโปรดักชันซิสเต็ม (production system) เครื่องกลนี้ได้จำลองแบบบนเครื่องคอมพิวเตอร์ธรรมดา โดยเขียนเป็นโปรแกรมในภาษาคอมพิวเตอร์ทั่วไป ตัวอย่างเช่น การจำลองแบบของโปรล็อกแมชีน ผลก็คือ โปรล็อกอินเตอร์พรีเตอร์ (Prolog Interpreter) สำหรับโปรแกรมระบบผู้เชี่ยวชาญ

โปรดักชันซิสเต็มเป็นโมเดลการคำนวณที่เสนอขึ้นเป็นครั้งแรกโดย Post (ผู้ออกแบบ Post Machine ผู้อ่านที่สนใจอาจศึกษารายละเอียดเกี่ยวกับโมเดลของการคำนวณแบบต่าง ๆ ได้จากหนังสือเกี่ยวกับทฤษฎีการคำนวณ : Theory of Computation) ตั้งแต่ปี 1943 วัตถุประสงค์เพื่อให้เป็นโมเดลของการคำนวณทั่ว ๆ ไป โมเดลนี้มีความสามารถเท่าเทียมกับเครื่องกล Turing ซึ่งเป็น

โมเดลการคำนวณอีกแบบหนึ่งที่ได้รับความนิยมมากจนกลายเป็นโมเดลการคำนวณของคอมพิวเตอร์ในปัจจุบัน เพราะเราสามารถใช้โปรดักชันซิสเต็มทำงานทุกอย่างที่เครื่องคอมพิวเตอร์ในปัจจุบันทำได้ ขณะเดียวกันเนื่องจากภาษาคอมพิวเตอร์ทั่วไปแบบฟอน นอยแมนได้ผ่านการพิสูจน์แล้วว่า มีความสามารถเท่าเทียมกับเครื่องกล Turing ดังนั้นจึงกล่าวได้ว่า เราสามารถใช้โปรดักชันซิสเต็มทำทุกอย่างที่ภาษาคอมพิวเตอร์ทั่วไปทำได้ ด้วยเหตุนี้โปรดักชันซิสเต็มจึงเปรียบเสมือนกับเป็นภาษาคอมพิวเตอร์อีกภาษาหนึ่ง แนวความคิดนี้ได้รับความสนใจจากนักคอมพิวเตอร์ไม่น้อย ได้มีการพัฒนาโปรดักชันซิสเต็มขึ้นเป็นภาษาคอมพิวเตอร์จริง ๆ นั่นคือ ภาษา OPS5

Production System

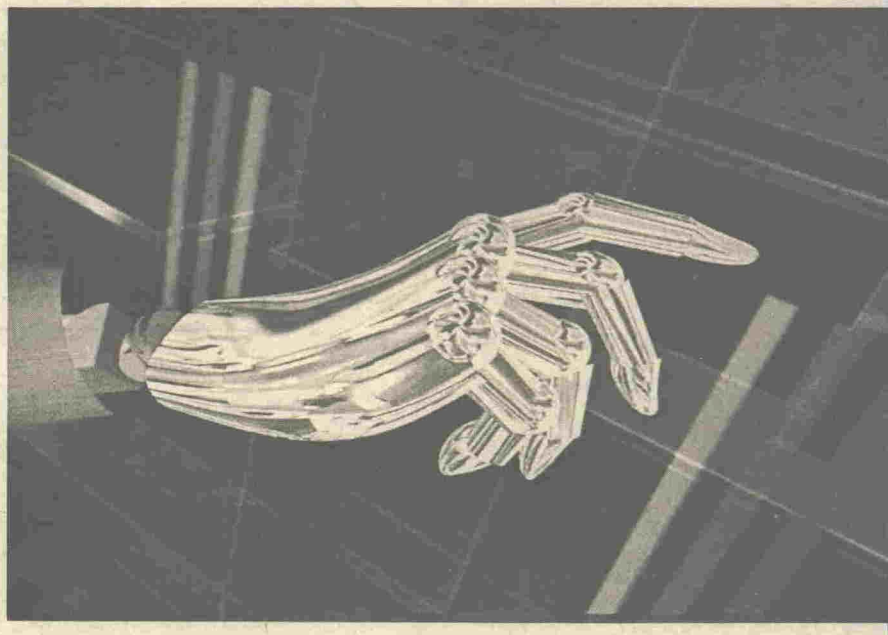
กลไกของโปรดักชันซิสเต็มประกอบด้วย

1. ชุดคำสั่ง ซึ่งแต่ละคำสั่งจะอยู่ในรูป implication คือมีเงื่อนไขอยู่ทางซ้ายและปฏิบัติการอยู่ทางขวาเช่นเดียวกับคำสั่ง IF...THEN...

2. ฐานข้อมูล สำหรับบันทึกสถานะของการปฏิบัติการ หรืออีกนัยหนึ่งคือส่วน "ความคิด" ในโมเดลข้างต้น และจะตรงกับส่วน "ความจำชั่วคราว" ในโครงสร้างของโปรแกรม

3. กลไกการเรียกคำสั่งมาปฏิบัติ เป็นลูปที่เลือกคำสั่งที่สามารถปฏิบัติได้มาปฏิบัติ วนอยู่อย่างนี้จนจบโปรแกรม ส่วนนี้จะตรงกับกลไกควบคุมการอนุมานในโครงสร้างโปรแกรม

ผู้เขียนได้สมมติโปรแกรมเลียนแบบตัวอย่างของ Allen Newell (ผู้เขียนโปรแกรม General Problem Solver



เป็นผู้บุกเบิกงานทางปัญญาประดิษฐ์คนหนึ่ง) แต่ได้ทำให้ง่ายขึ้นเพื่อมองเห็นชัด ๆ โดยจะอธิบายการทำงานควบคู่กันเลย โปรแกรมของเราในที่นี้มีเพียงสองคำสั่งดังนี้

```
1. (ดู า ---> ส า)
2. (ข ---> ดู)
```

คำสั่งที่หนึ่งกล่าวว่า ถ้าในฐานข้อมูลมีคำว่า "ดู" และสระ "า" ให้แทนด้วยอักษร "ส" และสระ "า" ไว้ข้างหน้า คำสั่งที่สองกล่าวว่า ถ้าในฐานข้อมูลมีอักษร "ข" ให้แทนด้วยคำว่า "ดู" ไว้ข้างหน้า สมมติเรามีฐานข้อมูลเริ่มต้นดังนี้

```
ฐานข้อมูล : (ก ข า บิน มา)
```

จะเห็นว่ามีแต่คำสั่งที่ 2 ที่นำมาปฏิบัติได้คือ มีอักษร "ข" อยู่ในฐานข้อมูลจริง เพราะฉะนั้นปฏิบัติการแรกก็คือลบ "ข" ในฐานข้อมูลทิ้ง เอาคำว่า "ดู" ใส่ไว้ข้างหน้าแทน ถึงตอนนี้เราได้ฐานข้อมูลเป็น

```
ฐานข้อมูล : (ดู ก า บิน มา)
```

เมื่อปฏิบัติการคำสั่งแรกเสร็จแล้ว โปรแกรมก็กลับไปดูชุดคำสั่งใหม่ว่ามีคำสั่งใดทำได้อีก คราวนี้จะเห็นว่า คำสั่งที่ 2 ไม่เป็นจริงอีกต่อไป แต่คำสั่งที่ 1 นำมาปฏิบัติได้คือ มีคำว่า "ดู" และสระ "า" ในฐานข้อมูลจริง จึงนำคำสั่งที่ 1 มาปฏิบัติ เราจะได้ฐานข้อมูลเป็น

```
ฐานข้อมูล : (ส า ก บิน มา)
```

หลังจากนี้ลองวนกลับไปดูชุดคำสั่งใหม่ปรากฏว่าไม่มีคำสั่งใดที่จะนำมาปฏิบัติได้อีก โปรแกรมจึงยุติลง

กลไกของระบบผู้เชี่ยวชาญ

ในโปรแกรมระบบผู้เชี่ยวชาญส่วนใหญ่การทำงานจะคล้ายคลึงกัน อาจแตกต่างกันออกไปบ้างในรายละเอียดอยู่ 2 ประการได้แก่ ประการแรกคือลักษณะของคำสั่งและปฏิบัติการในคำสั่ง ประการที่สองเป็นกลไกในการเรียกคำสั่งมาปฏิบัติในแต่ละรอบ เรียกว่า control strategy อันที่จริงแม้กระทั่งโปรแกรมระบบผู้เชี่ยวชาญใหญ่ ๆ แต่ละระบบก็มีส่วนรายละเอียดเหล่านี้แตกต่างกันไปแทบทุกระบบ เนื่องจากมีวิธีการเล็ก ๆ น้อย ๆ เกี่ยวกับเทคนิคการเขียนโปร-

แกรมบางอย่างที่อาจจะเหมาะสมกับปัญหาหนึ่งแต่ไม่ตรงกับปัญหาอีกประเด็นหนึ่ง เช่น การใช้โครงสร้างข้อมูล (data structure) เป็นต้น แต่หลักการใหญ่ ๆ จะเหมือนกัน

ลักษณะคำสั่ง

คำสั่งในโปรแกรมระบบผู้เชี่ยวชาญมักอยู่ในรูปแบบเดียวคือ implication โดยมีเงื่อนไขอยู่ทางซ้ายเรียกว่า antecedent และมีข้อสรุปอยู่ทางขวา เรียกว่า consequent แต่ละคำสั่งอาจแบ่งตามวิธีทำงานได้สามแบบคือ

– คำสั่งที่เป็นหลักเกณฑ์ความรู้ การทำงานของคำสั่งแบบนี้ ถ้าเงื่อนไขทางซ้ายเป็นจริง เราจะได้ผลสรุปเป็นความรู้ใหม่จากทางขวาของคำสั่ง จัดว่าเป็นข้อสรุปความรู้จากผู้เชี่ยวชาญ ทั้งที่เป็นความรู้แบบหลักมาตรฐานและที่เป็น heuristic ผู้สร้างโปรแกรมต้องทำงานร่วมกับผู้เชี่ยวชาญเพื่อเขียนคำสั่งประเภทนี้ขึ้นมา

– คำสั่งประเภทปฏิบัติการ การทำงานของคำสั่งแบบนี้ ถ้าเงื่อนไขของคำสั่งทางซ้ายมือสามารถปฏิบัติได้ โปรแกรมจะปฏิบัติการทางขวามือ คำสั่งแบบนี้เป็นคำสั่งสำหรับการทำงานของโปรแกรมแบบเดียวกับคำสั่งโปรแกรมคอมพิวเตอร์ทั่ว ๆ ไป เช่น คำสั่งเพื่อการจัดการกับโครงสร้างข้อมูล

– คำสั่งที่มีลักษณะผสมของคำสั่งทั้งสองแบบ ทำให้โปรแกรมสามารถใช้ความรู้ที่ได้มาช่วยควบคุมงานภายในของโปรแกรมได้ เช่น ควบคุมการตอบคำถามของผู้ใช้

นอกจากนี้คำสั่งแต่ละประเภทยังมีหลายชนิด ยกตัวอย่างเช่น คำสั่งที่เป็นหลักเกณฑ์ความรู้อาจแบ่งเป็นความรู้ต่าง ๆ กันหลายจำพวก เช่น พวกที่เกี่ยวกับปัญหาที่จะแก้ไขโดยตรง โปรแกรมสำหรับวิเคราะห์โรคทางการแพทย์ โปรแกรมความรู้เกี่ยวกับการใช้ความรู้เหล่านี้อีกทีหนึ่ง (meta knowledge) เป็นต้น

ประเภทของคำสั่งที่สำคัญที่สุดในระบบได้แก่ คำสั่งประเภทความรู้ เนื่องจากเป็นคำสั่งที่เกี่ยวกับการแก้ปัญหาโดยตรง ในการปฏิบัติคำสั่งประเภทนี้แต่ละคำสั่งเป็นการทำงานตามหลักของการอนุมานคือ การ "รวบรวมหลักฐาน" ในการใช้เหตุผล การ "สรุปความรู้ใหม่" จากหลักฐานดังกล่าว และการ "จัดการกับความรู้ใหม่" ที่ได้ โดยทั่วไปแล้วปฏิบัติการทางซ้ายมือของคำสั่งจะเป็นส่วนของการรวบรวมหลักฐาน หลักฐานเหล่านี้จะถูกส่งให้กับกลไกการอนุมานเพื่อทำการประเมินสรุปความรู้ใหม่ ส่วนปฏิบัติการทางขวามือคือการจัดการกับความรู้ใหม่ที่สรุปได้จากการปฏิบัติตามคำสั่ง

การรวบรวมหลักฐานอาจได้จากการที่คำสั่งถามผู้ใช้ให้ป้อนข้อมูลแก่โปรแกรม หรือคำสั่งอาจเรียกเอาความรู้ที่อยู่ในฐานข้อมูล (ความจำชั่วคราว) มาใช้ หรือคำสั่งอาจใช้ความรู้ที่เป็นอัลกอริทึมคำนวณหาค่าต่าง ๆ ก็ได้ แต่การทำเช่นนี้ต้องไม่เป็นการทำให้หน่วยของความรู้ในฐานข้อมูลเปลี่ยนแปลง ส่วนการสรุปความรู้ใหม่จะเป็นไปตามกลไกของการใช้เหตุผล ซึ่งเป็นวิธีทางตรรกศาสตร์ ในส่วนของการจัดการกับความรู้ใหม่จะเป็นการเปลี่ยนแปลงฐานข้อมูล การเปลี่ยนแปลงอาจเป็นการเพิ่มหน่วยของความรู้ ลดหรือเปลี่ยนค่าที่มีอยู่ โดยปกติการเปลี่ยนแปลงฐานข้อมูลเป็นการ "รวบรวมความรู้" ขึ้นเป็นโครงสร้างที่มีความสัมพันธ์กันในความหมายของปัญหาที่กำลังทำอยู่ โครงสร้างนี้อาจจะอยู่ในรูปโครงสร้างข้อมูล (data structure) แบบต้นไม้ เช่น Context tree ในโปรแกรมมัยซิน หรืออาจเป็นรูปเน็ตเวอร์ก เช่น Semantic nets ในโปรแกรมปริซึม เป็นต้น

control strategy

การปฏิบัติงานทั้งหมดของโปรดักชันซิสเต็มในโปรแกรมระบบผู้เชี่ยวชาญจะอยู่ภายใต้การควบคุมของลูปใหญ่ ทำหน้าที่เลือกคำสั่งที่เหมาะสมมาปฏิบัติ เนื่องจากแต่ละคำสั่งจะทำงานได้ก็ต่อเมื่อเงื่อนไขทางซ้ายมือของคำสั่งเป็นจริง เราจึงเรียกการวนลูปนี้ว่า Recognize-act cycle ตัวอย่างเช่น recognize-act cycle ของ OPS5 มีอัลกอริทึม ดังนั้น

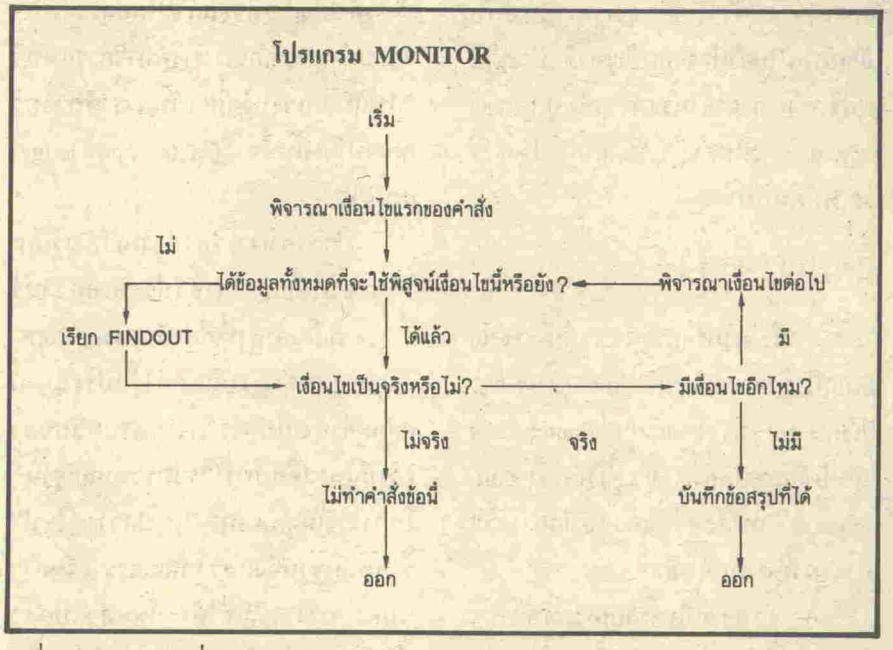
```
ลูป : match ซ้ายมือของคำสั่งทั้งหมดกับความ-
              รู้ในความจำชั่วคราว
      จบ    เมื่ออย่างหนึ่งอย่างใดต่อไปนี้
              ไม่มีคำสั่งทางซ้ายให้ match อีก
              ซ้ายมือเป็นคำสั่งหยุดการ match
              จำนวนรอบที่ตั้งไว้สิ้นสุด
              ถึงคำสั่งให้หยุดที่ตั้งไว้จากดีบัก-
              เกอร์
      แก้ปัญหาคำสั่งหลายคำสั่งสามารถปฏิบัติ
      ได้ในรอบเดียวกัน
      ปฏิบัติคำสั่งที่เลือกได้
      วนลูป
```

หัวใจของลูปนี้คือ การเลือกคำสั่งมาปฏิบัติ ซึ่งจะมีความเกี่ยวพันกัน 2 เรื่องคือ ตัวคำสั่งกับความรู้ในฐานข้อมูล เนื่องจากคำสั่งส่วนใหญ่จะปฏิบัติได้ก็ต่อเมื่อมีการพิสูจน์ความจริงของเงื่อนไขทางซ้ายมือ โดยอาศัยความรู้หรือหลักฐานในฐานข้อมูล (ความจำชั่วคราว)

เทคนิคการเลือกคำสั่งจากความรู้ในฐานข้อมูล

วิธีที่นิยมใช้ที่สุดได้แก่ การเลือกคำสั่งที่ใช้ความรู้ใหม่ล่าสุดอันเป็นผลมาจากการอนุมานครั้งสุดท้าย เพราะเป็นความรู้ที่เกี่ยวข้องกับเหตุผลที่โปรแกรมกำลังหาข้อสรุป ในทางปฏิบัติโปรแกรมอาจใช้วิธีให้ค่าเวลาที่ผ่านไปของการทำงานแก่ความรู้แต่ละอย่างในฐานข้อมูล ค่าเวลานี้นับเป็นจำนวนรอบของปฏิบัติการที่ได้มาซึ่งความรู้นั้น ๆ ตัวอย่าง หาก a, b, c และ d เป็นความรู้ในฐานข้อมูล เราได้ความรู้ a มาจากการทำงานรอบที่ 1 เราให้ค่าแก่ a เป็น "a (#1)" ค่า #1 ในวงเล็บคือเวลา ทำนองเดียวกันเราอาจได้ "b (#8)", "c (#4)" และ "d (#2)" สมมติว่าโปรแกรมมีคำสั่งสองข้อที่ใช้ได้ทั้งคู่ คำสั่งข้อที่หนึ่งต้องการความรู้ a และ b ส่วนคำสั่งข้อที่สองอาจต้องการความรู้ข้อ a, c และ d ในกรณีนี้เราจะเลือกคำสั่งที่ใช้ความรู้ a และ b ก่อน เนื่องจากเวลาของ a+b > a+c+d เป็นต้น

วิธีการเลือกคำสั่งที่จะนำมาปฏิบัติการมักใช้วิธีการเสนอ (propose method) มักใช้คู่กับการเลือกโดยหลักเกณฑ์บางอย่าง เช่น ความจำเพาะ (specificity) หรือความแน่นอน (certainty) วิธีการเสนอนั้นเป็นการเลือกกลุ่มของคำสั่งที่เกี่ยวข้องมาพิจารณา วิธีการนี้ใช้ฟังก์ชันที่เรียกตัวเอง (recursive function) สองฟังก์ชันทำงานร่วมกัน ฟังก์ชันหนึ่งเป็นมอนิเตอร์คอยตัดสินว่าคำสั่งใดควรจะทำต่อไป อีกฟังก์ชันหนึ่งทำหน้าที่เสนอกลุ่มของคำสั่งที่จะพิจารณาให้ทำต่อไป ในบรรดาคำสั่งที่เสนอมานี้ คำสั่งใดที่มีเงื่อนไขทางซ้ายมือมากที่สุด คำสั่งนั้นมีอภิสิทธิ์สูงกว่าคำสั่งอื่น ๆ (เงื่อนไขของความจำเพาะ) และขณะเดียวกันคำสั่งที่เป็นเหตุผลที่ชัดเจน เป็นความรู้ที่แน่นอนก็จะได้อภิสิทธิ์สูงเช่นกัน (เงื่อนไขของความแน่นอน) การให้น้ำหนักอภิสิทธิ์อาจใส่เป็นตัวเลขโดยวิธีใดก็ได้ ยกตัวอย่างอัลกอริทึมที่ใช้อยู่ในโปรแกรมมัยซินดังในรูปที่ 3



รูปที่ 3 กลไกการเลือกคำสั่งโดยวิธีเสนอ

สรุป

เรื่องที่เราจะทำความเข้าใจกับระบบผู้เชี่ยวชาญได้อย่างลึกซึ้งถ่องแท้นั้นคงทำได้ยาก เนื่องจากหลักเกณฑ์มากมายที่เป็นธรรมชาติของระบบผู้เชี่ยวชาญซึ่งมีพื้นฐานเดียวกับมันสมองของมนุษย์ ถ้าเราให้คำตอบได้ว่าสิ่งที่มนุษย์แสดงออกประกอบด้วยขั้นตอนเช่นใดบ้าง เราก็จะสามารถสร้างระบบ "ผู้เชี่ยวชาญ" ได้ ในบทความนี้ผู้เขียนพยายามตีแผ่ให้เห็นโครงสร้างทางสถาปัตยกรรมของระบบดังกล่าวอย่างชัดเจน โดยกล่าวถึงรายละเอียดกว้าง ๆ พอเป็นสังเขป เพื่อปูทางสู่การใช้ปัญญาประดิษฐ์ในระบบผู้เชี่ยวชาญในที่สุด

ส่วนตอนสองผู้เขียนจะนำเสนอเรื่องราวเกี่ยวกับการเขียนความรู้ในโปรแกรมระบบผู้เชี่ยวชาญในรูปของกฎเกณฑ์และกลไกของการใช้ปัญญาในโปรแกรมระบบผู้เชี่ยวชาญต่อไป ติดตามอ่านตอนจบในฉบับหน้า